The Tangle

Serguei Popov*

February 9, 2018. Version 1.4.2

概要

本論文では IOTA（IoT 産業のための暗号通貨）の数学的基礎を解析する．この革新的な暗号通貨の主たる特徴は，*Tangle*（大量のトランザクションのための有向非循環グラフ（DAG））である．Tangle は次の進化のステップとしてブロックチェーンを自然に継承し，M2M のマイクロペイメントシステムを確立するために必要な機能を提供する．

この論文の重要な貢献はマルコフ連鎖モンテカルロ（MCMC）アルゴリズムの族である．これらのアルゴリズムは Tangle に到着したばかりのトランザクションのために連結サイトを選択する．

1 システムの紹介と説明

過去 6 年間の Bitcoin の勃興と成功はブロックチェーン技術が現実社会において価値を持つことを証明するものであった．しかし，この技術には世界中の暗号通貨の一般的なプラットフォームとしての使用を妨げる多くの欠点もある．顕著な欠点の一つにはいかなる価値のトランザクションにも手数料が発生することである．急速に発展している IoT 産業では，マイクロペイメントの重要性が高まり，取引される価値よりも**大きな**手数料を支払うことは合理的でない．その上，ブロックチェーンインフラにおける手数料はブロック生成者のインセンティブとして働くので手数料を取り除くことは容易でない．これは既存の暗号通貨技術，すなわちシステムの異種混合性に別の問題を引き起こす．まずこのシステムにはトランザクションを発行するユーザーとトランザクションを承認するユーザーの異なるタイプの 2 種類のユーザーがいる．そしてこのシステムの設計では一部のユーザーへの避けられない優遇を引き起こし，全てのユーザーが論争解決に労力を費やす問題を次々に引き起こす．前述の問題は Bitcoin とその他多くの暗号通貨の基盤であるブロックチェーン技術とは本質的に異なる解決策を探る十分な根拠を示す．

この論文ではブロックチェーン技術を用いない革新的な手法について考察する．この手法は IoT 産業向けに特別に設計された *iota*[1] と呼ばれる暗号通貨に現在実装され

*a.k.a. `mthcl`; author's contact information: `e.monetki@gmail.com`

ている．この論文の目的は Tangle の一般的な特徴に焦点を当て，ブロックチェーンを取り除き，分散型台帳を維持しようとする時に起こる問題について考察する．また iota プロトコルの具体的な実装については考察しない．

一般的に Tangle に基づく暗号通貨は次のように働く．まずグローバルブロックチェーンではなく，*Tangle* と呼ばれる DAG がある．ノードによって発行されたトランザクションはトランザクションを保存するための台帳である Tangle グラフのサイト集合を構成する．Tangle のエッジ集合は次のようにして得られる：新しいトランザクションが到着すると，前の 2 つ[1]のトランザクションを**承認する**必要がある．これらの承認は図 1[2]に示すように，有向エッジ（≒矢印）によって表される．トランザクション A とトランザクション B との間に有向エッジがなく，A から B までに少なくも 2 の長さの有向パス（$A \to X \to B$ となる道順）がある場合，A は B を**間接的に承認**するという．また他のすべてのトランザクションによって間接的または直接的に承認される“ジェネシス”トランザクションも存在する（図 2 を参照）．ジェネシスは以下のように記述される．Tangle の初めにおいて，すべてのトークンを含むバランスのとれた 1 つのアドレスが存在した．そしてジェンシストランザクションはそのすべてのトークンを他のいくつかの“創設者”アドレスに送った．すべてのトークンがジェネシストランザクションで作られたことを強調しておこう．よって将来的にトークンは生成されず，採掘者が“どこからともなく”報酬を受け取るという意味での採掘はない．

用語についての簡単な注記：**サイト**とは Tangle グラフ上にあるトランザクションのことである．ネットワークは各ノードによって構成されており，各ノードはトランザクションを発行し検証する実在物である．

Tangle の重要なアイデアは，トランザクションを発行するためにはユーザーは別のトランザクションを承認する必要があるということである．従って，トランザクションを発行するユーザーはネットワークセキュリティに貢献することになる．各ノードは到着したトランザクションが矛盾していないかどうかをチェックするものと仮定される．トランザクションが Tangle 履歴と矛盾するとノードが判断した場合，ノードは直接的または間接的に矛盾するトランザクションを承認しない[3]．

トランザクションが追加の承認を受け取ると，それはより高いレベルの信頼性でシステムによって受け入れられる．言い換えれば，このシステムが二重支払いのトランザクションを受け入れるようにすることは困難であるということである．ここでノードは承認するトランザクションを選択するためのいかなるルールをも課されていないことに注目することは重要である．その代わり，多くのノードがいくつかの“参照”ルールに従う場合，いかなる不正工作されたノードも同じ種類のルールに従うことがより良いと主張する[4]．これは妥当な仮定であり，各ノードがファームウェアを事前にイン

[1] これは最もシンプルな手法である．トランザクションが一般的な $k \geq 2$ となる k 個の別のトランザクションを承認する必要がある類似のシステムや全く異なったルールを持つシステムも研究されたい．

[2] 各図において時間は常に左から右にいくに従って増加する．

[3] もしあるノードが矛盾するトランザクションを承認する新しいトランザクションを発行すると，他のノードがその新しいトランザクションを承認せず，忘れ去られるリスクを負うことになる．

[4] この点については 4.1 節の最後で詳しく説明する．

ストールされた専用チップである IoT のコンテクストにおいては特に妥当な仮定であると言える．

ノードはトランザクションを発行するために，以下のことを行う．

- ノードはあるアルゴリズムに従って，承認する 2 つの他のトランザクションを選ぶ．一般的に，この 2 つのトランザクションは一致することがある．
- ノードは 2 つのトランザクションが矛盾していないかどうかを確認し，矛盾しているトランザクションは承認しない．
- ノードが有効なトランザクションを発行するには，Bitcoin のブロックチェーンと同様の暗号パズルを解く必要がある．これは承認されたトランザクションのいくつかのデータと鎖状に繋がったナンスのハッシュがある特定の形式を持つようにナンスを見つけることで達成される．Bitcoin プロトコルの場合はハッシュの先頭に少なくともあらかじめ決められた個数の 0 が並ぶようにナンスを見つける必要がある．

iota ネットワークが非同期であることは重要である．一般に各ノードはトランザクションの同じセットを見るとは限らない．また Tangle には矛盾しているトランザクションも含まれることに気づく必要がある．各ノードは有効な[5]トランザクションが台帳に入る権利があるかどうかの合意を得る必要はない．つまりすべてのトランザクションが Tangle に入れるということである．しかし，矛盾したトランザクションが存在する場合には，各ノードはどのトランザクションが孤立するかを決める必要がある[6]．各ノードが 2 つの矛盾するトランザクション間の決定に使う主なルールは次のとおりである：ノードはチップ選択アルゴリズム[7]（4.1 節を参照）を何度も実行し，2 つのうちどちらのトランザクションが，後からやってくるチップによって間接的に承認される可能性が高いかを見る．例えば，チップ選択アルゴリズムが 100 回実行されている間にあるトランザクションが 97 回選択された場合には，97%の信頼性で確立されたとする．

また，次の質問についてコメントしたい ([4] を参照)：「各ノードがトランザクションを増殖させる動機付けはどこにあるのか？」全てのノードがいくつかの統計情報を計算しており，そのうちの一つは隣人のノードから受け取った新しいトランザクションの数である．ある特定のノードが“怠けすぎ”の場合，隣人のノードによって外される．従ってノードがトランザクションを発行しないため，自身のトランザクションを承認する新しいトランザクションを共有する直接のインセンティブはないが，依然として参加するインセンティブがある．

[5]規約に従って発行されるトランザクション

[6]孤立したトランザクションはこれ以上新たに入ってきたトランザクションによって間接的に承認されることはない．

[7]上記で述べたように，他のノードがチップ選択のために同じアルゴリズムに従うと仮定する十分な理由がある．

2 章でいくつかの表記法を導入した後，どの 2 つのトランザクションを承認するかを選択するアルゴリズム，トランザクション承認全般を測定する規則 (3 章，特に 3.1 節)，そして起こりうる攻撃シナリオ (4 章) について議論する．また，数式に恐れを抱く（考えにくいが）場合，読み手は数式などを読み飛ばして各章の末尾の “結論” に直行することも出来る．

暗号通貨の世界において DAG を使うというアイデアはしばらく前からあったことは付言されるべきである（[3, 6, 7, 9, 12] を参照）．具体的には，[7] は GHOST プロトコルを導入し，これは Bitcoin プロトコルの修正版であり，主台帳をブロックチェーンの代わりにツリーにしたものである．このような変更は，確認時間を短縮し，ネットワークの全体的なセキュリティを向上させることが示されている．[9] の著者達は，DAG ベースの暗号通貨モデルを検討している．そのモデルは Tangle と以下の理由により異なる：その DAG のサイトは個々のトランザクションではなくブロックであり，システムの採掘者はトランザクションの手数料のために競争し，新しいトークンはブロックの採掘者によって作られる．また，[6] で Tangle と幾分類似した解決策が提案されていることに気づくが，特定のチップの承認戦略については考察されていない．この Tangle についての論文の最初の版が発行された後，[8] などの DAG ベースの分散型台帳を作成することを目指す他のいくつかの取り組みが登場した．また，P2P 決済チャンネルを確立することで Bitcoin における少額取引を可能にする別の取り組み（[2, 10]）も参照されたい．

2 荷重 等

この章でトランザクションの荷重と関連の概念を定義する．トランザクションの荷重は，トランザクションを発行するノードがそのトランザクションに投資した仕事量に比例する．現在の iota の実装では，荷重は 3^n を取ることしかできず，ここで n は許容可能な値の非空な区間に属する正の整数である[8]．実際問題としては荷重がどのように得られたかを知ることは重要ではない．すべてのトランザクションに正の整数の荷重がつけられていることだけが重要である．一般に，より大きな荷重を持つトランザクションがより小さな荷重を持つトランザクションよりも “重要” であるという考えである．スパムやその他の攻撃の型を避けるために，実在物は短期間で “許容可能な” 荷重を有する大量のトランザクションを生成することができないと想定される．私達にとって必要な概念の一つは，トランザクションの**累積荷重**である．これは特定のトランザクション自身の荷重に加え，直接的または間接的にそのトランザクションを承認する全てのトランザクションの自身の荷重の合計として定義される．この累積荷重の計算アルゴリズムは図 1 に示されている．各ボックスはトランザクションを表し，各ボックス内右下隅の数字はトランザクション自身の荷重を示し，太字の数字は累積荷重を示す．例えば，トランザクション F は，トランザクション A，B，C，E によって直接的または間接的に承認される．トランザクション F の累積荷重は $9 = 3 + 1 + 3 + 1 + 1$

[8] この区間は有限でもあるべきである—4 章の “大きな荷重攻撃” を参照のこと

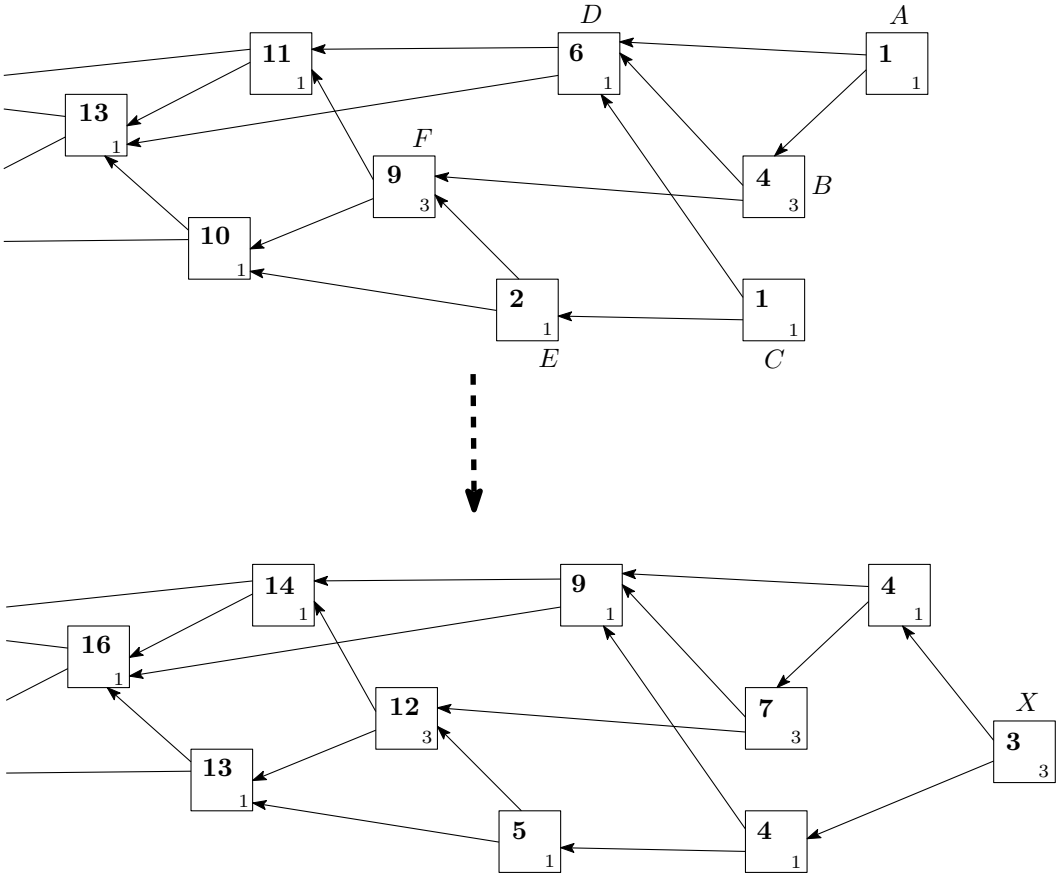

図 1: 新たに発行されたトランザクション X の荷重が割り当てられる前と後の DAG. 各ボックスはトランザクションを表し，右下の小さな数字はトランザクション自身の荷重を表し，太字は累積荷重を表す.

(F 自身の荷重と A，B，C，E それぞれの自身の荷重の合計）となる.

Tangle グラフにおいて承認されていないトランザクションを“チップ”と定義する．図 1 の上部にある Tangle の概略では A と C だけがチップである．図 1 の下部にある Tangle の概略では，新たなトランザクション X が現れ A と C を承認するので，X が唯一のチップとなる．ここで他のすべてのトランザクションの累積荷重は X 自身の荷重である 3 だけ増加する.

承認アルゴリズムのために 2 つの追加の変数を導入する必要がある．初めに Tangle 上のトランザクションサイトのために,

- **高さ**：ジェネシスへ方向付けられた最長のパスの長さ,

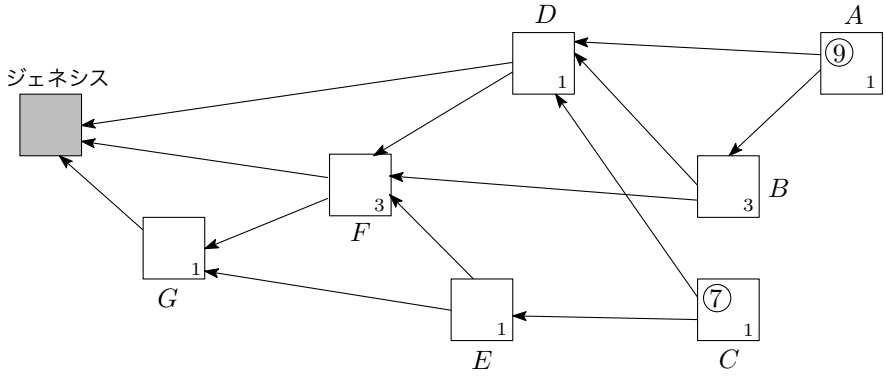

図 2: 各サイトに自身の荷重が割り当てられた DAG，そしてサイト A と C のスコアを計算した．

- **深さ**：どれかのチップへ（つまりジェネシスとは逆の方向に）方向付けれられた最長のパスの長さ，

を導入する．

例えば，図 2 で G は 1 の高さで，逆経路 F，D，B，A のために 4 の深さを持ち，D は 3 の高さで 2 の深さである．また，**スコア**と言う概念を導入する．トランザクションのスコアはそのトランザクションによって直接的にも間接的にも承認された全てのトランザクションの荷重と，それ自身のトランザクションの荷重との合計として定義される．

図 2 において，A と C だけがチップである．トランザクション A は直接的または間接的にトランザクション B, D, F, G を承認するので, A のスコアは $1+3+1+3+1 = 9$ となる．同様に C のスコアは $1+1+1+3+1 = 7$ となる．

この論文で掲示された議論を理解するために，すべてのトランザクション自身の荷重が 1 と仮定しても差し支えない．**以降，この仮定のみ考慮する．**この仮定の下では，トランザクション X の累積荷重は直接的または間接的に X を承認するトランザクションの数に 1 を加えたもので，スコアは X に直接的または間接的に承認されたトランザクションの数に 1 を加えたものである．

このセクションで定義されているものの中で，累積荷重が（断然）最も重要な測定基準であることに注意して頂きたい．ただし，高さ，深さ，スコアもいくつかの議論に一時的に入る．

3　システムの安定性とカット集合

時点 t におけるシステムのチップの合計数を $L(t)$ とする．確率過程 $L(t)$ は**安定**していると期待される[9]．より正確には，プロセスが**正再帰**であると期待され，正式な定義については [11] の 4.4 節と 6.5 節を参照されたい．特に，正再帰は $\mathbb{P}[L(t) = k]$ as $t \to \infty$ の極限が存在すべきで，全ての $k \geq 1$ について正であるべきことを暗に意味する．直観的に，$L(t)$ が定数のあたりで変動し，無限にエスケープしないと期待する．もし $L(t)$ が無限にエスケープしたら，多くの承認されていないトランザクションが残ることになる．

$L(t)$ の安定性特性を分析するために，いくつの仮定を作る必要がある．1 つの仮定はトランザクションが多数のほぼ独立した実在物によって発行されることであり，そのため入力されるトランザクションのプロセスはポアソン点過程によってモデル化できる (例えば [11] の 5.3 節を参照)．λ をこのモデル化したポアソン過程における単位時間あたりのトランザクションの期待発生回数（速度）とする．単純化のため，この速度が時間的に一定であると仮定する．全てのデバイスがほぼ同等の演算処理能力を持つこと，デバイスがトランザクションを発行するために必要な計算を実行するために必要な平均時間を h と仮定する．そして，すべてのノードは次のように振る舞うことを**仮定**しよう：まず，ノードがトランザクション発行の際，2 つのチップをランダムに選択し承認する．一般に“誠実なノード”がこの戦略を適用するのは良いアイデアでは**なく**，これには多くの実用上の欠点があるためである．特に，“怠惰な”ノードや悪意のあるノードに対して十分な防御にならないからである (下の 4.1 節を参照のこと)．

であるが，分析が簡単であるためこの戦略を熟考することで，より複雑なチップ選択戦略のためのシステムの挙動について洞察を得ることができる．

次に，任意のノードがトランザクションを発行する瞬間に，実際の Tangle の状態を観測するのではなく，ちょうど h 時間単位前の Tangle の状態を観測するというさらに単純な仮定を設定する．これは，特に，時間 t で Tangle にアタッチするトランザクションが，時間 $t + h$ でネットワークに見えるようになることを意味する．また，チップの数は，時間的にほぼ静止したままであり，$L_0 > 0$ の数に集中していると仮定する．以下では，λ_0 と h の関数として L_0 を計算する．

与えられた時間 t において，私たちはおおよそ λh 個の“隠れたチップ”(時間 $[t-h, t)$ の間でタングルにアタッチし，ネットワークにはまだ見えないチップ）を持つことに気づく．また，概して r 個の“明らかになったチップ”（時間 $t - h$ より前にタングルにアタッチしたチップと時間 t の時にチップのままであるチップ）があり，$L_0 = r + \lambda h$ と仮定する．定常性（時間や位置によってその確率分布が変化しないという確率過程の性質）によって，私たちは，時間 $t - h$ でチップであったが，時間 t においてはもうチップではない λh 個のサイトがあると仮定してもよい．ここで，この瞬間にやってくる新しいトランザクションについて考える．この時，承認することを選んだトランザクションは確率 $r/(r + \lambda h)$ を有するチップである（トランザクションを発行したノー

[9] プロセスが時間的に均一（斉時的）であるという追加の仮定に基づく．

ドには既知の約 r 個のチップがあり，さらにもうチップではない約 λh 個のトランザクションがあるがノードはチップと認識するため)．だから，選択されたチップの平均数は $2r/(r+\lambda h)$ である．注目すべき点は，定常性の体制において，選択されたチップの平均数は 1 に等しくなければならない．これは定常性の体制においては，平均的に新しいトランザクションはチップの数を変更してはならないからである．r について，方程式 $2r/(r+\lambda h)=1$ を解くと，$r=\lambda h$，ゆえに，

$$L_0 = 2\lambda h. \tag{1}$$

新しいトランザクションが 2 個の代わりに k 個のトランザクションを参照するルールである場合，同様の計算によって，

$$L_0^{(k)} = \frac{k\lambda h}{k-1}. \tag{2}$$

もちろん，この式は $L_0^{(k)}$ が $k \to \infty$ の時，λh に近づくという事実と一致している(基本的に，この λh 個のチップだけはまだネットワークにとって未知のものである)．

また，(2 個のトランザクションを承認するケースに戻る）トランザクションが最初に承認される予想時間は，およそ $h + L_0/2\lambda = 2h$ である．これは，私たちの仮定によれば，最初の時間単位 h の間にトランザクションは承認されることはないが，その後，それに対する承認のポアソン過程は約 $2\lambda/L_0$ の速度を持つからである．(起こりうるサブタイプのリストに従って，ポアソン過程の各イベントを独立して分類すると，各サブタイプのイベントの過程は独立したポアソン過程であると [11] の命題 5.3 は示していることを思い出して頂きたい．)

任意の固定時間 t において，$s \in [t, t+h(L_0, N)]$ のときにチップであったトランザクションの集合が一般的に**カット集合**を構成する[10]ことを観察されたい．時間 t より後に発行されたトランザクションからジェネシスへの任意のパスがこの集合を通過しなければならない．Tangle における新しいカット集合のサイズが時々小さくなることは重要である．この小さなカット集合を DAG 剪定（せんてい）やその他のタスクのためのチェックポイントとして利用することが出来る．

これは観察するのに重要なことであるが，上記の“純粋にランダムな”承認戦略は実際の場面においてあまり良いものとはいえない．なぜなら，チップの承認を奨励しないからである．“怠けたい”ユーザーはかなり古い特定のペアのトランザクションを常に承認することができ，よって最近のトランザクションの承認に貢献せず，そのような行動によって罰せられることもない[11]．また，悪意のあるものは，特定のペアのトランザクションを承認する多くのトランザクションを発行することによって，チップの数を人工的に膨らませることができる．新たなトランザクションが非常に高い確率で悪意のあるチップを選択し，“誠実なノード”に属するチップを事実上放棄するように

[10] 少なくともノードがチップを承認しようと**試みる**場合．

[11] 私達が特定のチップ選択戦略を**押し付け**ようとしていないことにご注意いただきたい．つまり攻撃者は都合の良い方法でチップを選択することができる．

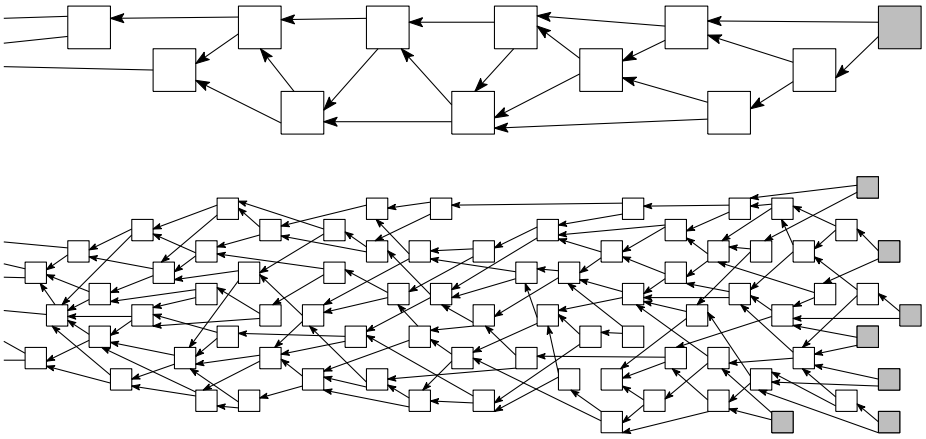

図 3: トランザクションフローの低負荷（上）と高負荷（下）の状態．白い四角は検証済みのサイトを，灰色の四角はチップを表す．

することができる．この種の問題を避けるためには，「より良い」チップに対して偏った戦略を採用する必要がある．そのような戦略の 1 つの例は，以下の 4.1 節に示さる．

トランザクションが初めての承認を受け取るのにかかると予想される時間について議論を始める前に，2 つの状態に区別できる事を言及する（図 3 を参照）．

- 低負荷：正常なチップの数が小さく，頻繁に 1 になる．これはトランザクションのフローが非常に小さく，複数の異なるトランザクションが同じチップを承認する可能性が低い時に起こりうる．また，ネットワークの通信遅延がとても短く，各デバイスの演算が高速な場合，多数のチップが現れる可能性は低い．これはトランザクションの流れが分別を持って大きい場合にも当てはまる．さらに人工的にチップの数を膨らませようとする攻撃者がいないことを前提としなければならない．

- 高負荷：正常なチップの数が大きい．トランザクションのフローが十分に大きく，ネットワークの通信遅延とともに演算遅延が発生し，複数の異なるトランザクションが同じチップを承認する可能性が高い時に起こりうる．

この分類は略式的なものであり，2 つの状態の間に明確な境界線はない．それでもこれらの極端に異なる 2 つの状態を考えることは示唆に富むと思われる．

低負荷状態における状況は比較的簡単である．最初の入ってきた少数のトランザクションのうちの 1 つが特定のチップを承認するので，最初の承認は位数 λ^{-1} の平均時間で起こる．

ここで L_0 が大きい高負荷な状態を考えよう．上述したように，異なるチップへの承認のポアソン過程は独立しており，およそ $2\lambda/L_0$ の速度を有すると仮定してよい．した

がって, トランザクションが最初に承認を受け取るまでの予想時間は, 約 $L_0/(2\lambda) \approx 1.45h$ (式 (1) により) である.

しかし, より精妙な承認戦略[12]のために, トランザクションが他から承認されるまで多くの時間を受動的に待つのは良い考えでないことは注目に値する. これは“より良い”チップは現れ続け, 承認において優先されるという事実からである. むしろ, トランザクションが承認を待っている時間が $L_0/2\lambda$ よりもはるかに長い場合は, 追加の空のトランザクションでこの承認を待っているトランザクションを促進することが良い戦略であろう[13]. つまり, ノードは, より良いチップの 1 つとともにその前のトランザクションを承認する空のトランザクションを発行することができる. こうすることで空のトランザクションが承認を受ける確率が高まる.

今ここで, 高さとスコアに基づく承認戦略は特定のタイプの攻撃に対して脆弱であるかも知れないことが明らかになっている (4.1 節を参照). 私達はそのような攻撃に対する防御のより精巧な戦略[14]について 4.1 節で議論する. その間, 新たに入ってくるトランザクションが 2 つのダンラムなチップを承認する簡単なチップ選択戦略を検討することはまだ価値がある. この戦略は分析が容易で, 従って Tangle の定性的および定量的な振る舞いをについて幾らかの洞察が得られるからである.

結論：

1. 私たちは, 2 つの状態, 低負荷と高負荷を区別する (図 3).
2. 低負荷状態ではごく少数のチップしかない. チップは単位時間 $\Theta(\lambda^{-1})$ で初めて承認を得る. ここで λ は入ってくるトランザクションのフローの速度である.
3. 高負荷型において, チップの典型的な数は新しいトランザクションによって採用されるチップ承認戦略に依存する.
4. トランザクションが 2 つのランダムなチップを承認する戦略を用いるならば, チップの典型的な数は式 (1) によって与えられる. この戦略はチップの典型的な数に関して適切であることが示され得る. ただし, この戦略の採用は実用性に欠ける. なぜならチップの承認を奨励しないからである.
5. 攻撃やその他のネットワークの問題を処理するためにより精妙な戦略が必要である. そのような一連の戦略が 4.1 節で説明される.

[12] iota の今後の実装では, “より良い”品質のチップに有利になる.

[13] 空のトランザクションとはトークンの移転を伴わないトランザクションであるが, 依然として 2 つの異なるトランザクションを承認する必要がある. 空のトランザクションを生成するとネットワークセキュリティが向上することに注意していただきたい.

[14] 実際, 著者の思いはチップ承認戦略こそが Tangle ベースの暗号通貨構築のための**最も重要な要素**であるということである. そこにいくつも攻撃ベクトルが隠れている. また, 特定のチップ承認戦略を強制する方法がないので, 各ノードが, 他ノードの多数が従うことを知った上で自発的にそうするようなものでなければならない.

6. 高負荷型において，チップが承認されるまでの典型的な時間は $\Theta(h)$ である．ここで h はノードの平均計算/伝播時間である．ただし，上記の時間内に最初の承認が起こらなかった場合，追加の空のトランザクションでトランザクションを促進するのは発行者/受領者の両者か一方かにとって良いアイデアである．

3.1 通常どれくらい早く累積荷重が成長するか?

ネットワークが低負荷状態であると仮定する．あるトランザクションが何度か承認された後，その累積荷重は λ のスピードで成長する．なぜなら，すべての新しいトランザクションがこのトランザクションを間接的に参照するからである[15]．

ネットワークが高負荷状態のケースでは，大きな累積荷重を持つ古いトランザクションは λ のスピードで累積荷重が成長する．なぜなら，基本的にすべての新しいトランザクションはその古いトランザクションを間接的に参照するからである．さらに，トランザクションが初めて Tangle に追加された時，承認されるために少し待たなければならないかもしれない．この時間間隔では，トランザクションの累積荷重は乱雑に振る舞う．トランザクションが複数の承認を受けた後に累積荷重が増加するスピードを特徴付けるために，$H(t)$ を時間 t における累積荷重の期待値（簡単にするために，私たちのトランザクションがネットワークに公開された瞬間から時間を数え始める．すなわち，トランザクションが作成された後の単位時間 h からである．），$K(t)$ を時間 t におけるトランザクションを承認するチップの期待数と定義しよう．$h := h(L_0, N)$ という短縮表記も用いる．時間内に，全体のチップの数がほぼ定数 L_0 に等しいという単純化された仮定も行う．この章では“2 つのチップをランダムに承認する”という戦略を用いて議論する．定性的な動作は他の妥当な戦略とほぼ同じ結果になると期待される．

時間 t の時点でシステムに到来するトランザクションが，システムの状態に基づき時間 $t-h$ の時点で，通常 2 つの承認するトランザクションを選択することを思い出して頂きたい．なぜならノードは実際にトランザクションを発行する前に何らかの計算と確認をせねばならないからである．Tangle において“私たちの”2 個のチップの少なくとも 1 個をトランザクションが承認する確率が $1-\big(1-\frac{K(t-h)}{L_0}\big)^2 = \frac{K(t-h)}{L_0}\big(2-\frac{K(t-h)}{L_0}\big)$[16] であることに気づくのは難しくない（ただし，$K(\cdot)$ は予想されるチップの数ではなく，**実際の**チップの数であると仮定する．）．例えば [11] の例 6.4 と相似して，小さい $\lambda > 0$ に対して，次のように書くことが出来る．

$$H(t+\delta) = H(t) + \lambda\delta\frac{K(t-h)}{L_0}\Big(2-\frac{K(t-h)}{L_0}\Big) + o(\delta).$$

従って次の微分方程式を演繹できる．

$$\frac{dH(t)}{dt} = \lambda\frac{K(t-h)}{L_0}\Big(2-\frac{K(t-h)}{L_0}\Big). \tag{3}$$

[15] すべてのトランザクションの自身の荷重を 1 と仮定しているので，累積荷重はトランザクションを直接的にまたは間接的に参照したトランザクションの数に 1 を加えただけのものになる．

[16] 左辺の式は承認された 2 つのチップが私達のものでない確率を 1 から引いたものである．

式 (3) を利用できるためには，まず $K(t)$ を計算する必要がある．$t-h$ の時点でのチップが t の時点でのチップではない可能性があるため，これは自明な作業でない．入ってきたトランザクションがそのようなチップを承認する場合，元のトランザクションを承認するチップの全体数は 1 増加する．極めて重要な観察は，$t-h$ の時点のチップが t の時点でチップであり続ける確率は約 $\frac{1}{2}$ である．（これを確認するには 3 章の説明を思い出して頂きたい．チップの典型的な数は $2\lambda h$ 個であり，期間 h の間隔の間，新しい λh 個のチップは古いチップの半分と置き換わる．）従って，t の時点で約 $K(t-h)/2$ のチップが未確認のチップ状態にとどまり，残りの半分は少なくとも 1 つの承認を受けるだろう．ここで A は t の時点でもまだチップである $t-h$ の時点での $K(t-h)/2$ のチップ集合を示し，B は t の時点ではもうすでに承認された $K(t-h)/2$ の残りのチップ集合を示すとする．仮に p_1 を新しいトランザクションが B から少なくとも 1 つのトランザクションを承認し，A からは全くトランザクションを承認しない確率とする．また，仮に p_2 を両方の承認されたトランザクションが A に属する確率とする．言い換えれば，p_1 と p_2 はそれぞれ“私たちの”チップの現在の数が新たなトランザクションが到来した際に 1 増加する，または 1 減少する確率である．つまり，

$$p_1 = \Big(\frac{K(t-h)}{2L_0}\Big)^2 + 2\times\frac{K(t-h)}{2L_0}\Big(1-\frac{K(t-h)}{L_0}\Big),$$
$$p_2 = \Big(\frac{K(t-h)}{2L_0}\Big)^2.$$

最初の式を得るには，p_1 が，([両方の承認されたチップが B に属する確率]+[1 つ目のチップが B に属し 2 つ目のチップが $A\cup B$ に属さない確率の 2 倍]）に等しいことを観察する．式 (3) と類似して， $K(t)$ の微分方程式は

$$\frac{dK(t)}{dt} = (p_1-p_2)\lambda = \lambda\frac{K(t-h)}{L_0}\Big(1-\frac{K(t-h)}{L_0}\Big). \tag{4}$$

式 (4) を厳密に解くのは難しいので，更に単純化された仮定を行う．まず，私達は $K(t)$ が特定の $\varepsilon>0$ について εL_0 のレベルに到達した時，$(1-\varepsilon)L_0$ までかなり早く成長すると気付く．さて，$K(t)$ が L_0 に対して小さい時，私達は式 (4) の右側の最後の係数を落とすことができる[17]．$\frac{\lambda h}{L_0}=\frac{1}{2}$ であることを思い出し，式 (4) を簡略化したヴァージョンを得る：

$$\frac{dK(t)}{dt} \approx \frac{1}{2h}K(t-h). \tag{5}$$

ここで境界条件は $K(0)=1$ である．$K(t)=\exp(c\frac{t}{h})$ の形の解を考えるため，式 (5) に $K(t)=\exp(c\frac{t}{h})$ を代入して，次式が得られ，

$$\frac{c}{h}\exp\Big(c\frac{t}{h}\Big) \approx \frac{1}{2h}\exp\Big(c\frac{t}{h}-c\Big),$$

[17] 1 に近い定数になるので右辺は次のようになる $\lambda\frac{K(t-h)}{L_0}$.

よって，次式が近似解となる．

$$K(t) = \exp\left(W\left(\tfrac{1}{2}\right)\frac{t}{h}\right) \approx \exp\left(0.352\frac{t}{h}\right). \tag{6}$$

ここで，$W(\cdot)$ はランベルト W 関数[18]と呼ばれるものである．よって，式 (6) において両辺の対数をとると，$K(t)$ が εL_0 に達する時間はおおよそ次式であることが分かる．

$$t_0 \approx \frac{h}{W\left(\frac{1}{2}\right)} \times (\ln L_0 - \ln \varepsilon^{-1}) \lesssim 2.84 \cdot h \ln L_0. \tag{7}$$

式 (3) に戻り，右側の最後の係数を落とす．私達は「適応期間」の間（すなわち t_0 が式 (7) の場合 $t \le t_0$），次が成立することを得る．

$$\begin{aligned}\frac{dH(t)}{dt} &\approx \frac{2\lambda}{L_0}K(t-h)\\ &\approx \frac{1}{h\exp\left(W(\frac{1}{2})\right)}\exp\left(W\left(\tfrac{1}{2}\right)\frac{t}{h}\right)\\ &= \frac{2W\left(\frac{1}{2}\right)}{h}\exp\left(W\left(\tfrac{1}{2}\right)\frac{t}{h}\right)\end{aligned}$$

従って，

$$H(t) \approx 2\exp\left(W\left(\tfrac{1}{2}\right)\frac{t}{h}\right) \approx 2\exp\left(0.352\frac{t}{h}\right). \tag{8}$$

読者には，適応期間の後，累積荷重 $H(t)$ は λ のスピードで線形的に増加することを思い出していただきたい．私達は式 (8) にあるような“指数関数的成長”は，適応期間の間に累積荷重が“とても早く”成長することを意味しないことを強調したい．むしろ，その振る舞いは図 4 に示すとおりである．

結論：

1. トランザクションが低負荷状態で複数回承認された後，その累積荷重は λw のスピードで成長していく．ここで w は一般的なトランザクションの荷重平均である．

2. 高負荷状態では，2 つの異なる成長段階が存在する．まず初めに，トランザクションの累積荷重 $H(t)$ は**適応期間**の間，式 (8) による増加速度で増加する．適応期間が終了すると，累積荷重は λw の速さで増加する（図 4 を参照）．実際，**すべての**合理的な戦略では，適応期間の終了後，累積荷重はこのスピードで増加していく．なぜなら，新たに入ってくるトランザクションは利益のあるトランザクションを間接的に承認するからである．

[18]オメガ関数もしくは対数積として知られている．$x \in [0, +\infty)$ の時，$x = W(x)\exp(W(x))$ の特性を持つ．

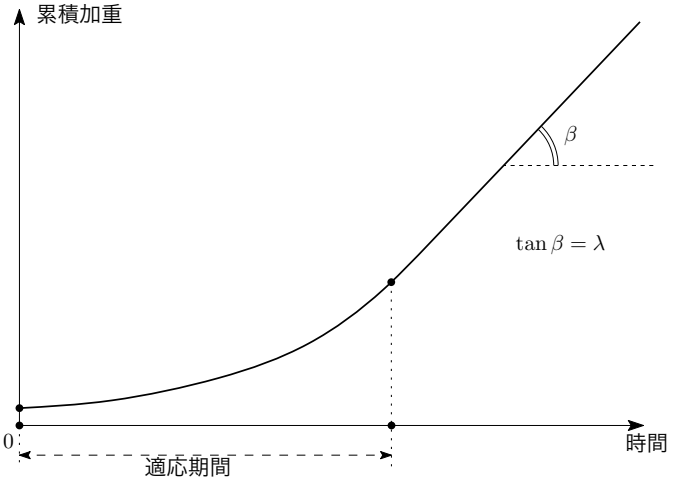

図 4: 高負荷状態における累積荷重と時間のプロット.

3. あるトランザクションの適応期間は現在のチップのほとんどがそのトランザクションを間接的に承認するまでの時間として考えることができる．適応時間の典型的な長さは式 (7) によって与えられる.

4 起こりうる攻撃シナリオ

私たちは攻撃者が単体でネットワークを“上回る”ように試みる攻撃シナリオについて説明する.

1. 攻撃者は商人に支払いを送り，トランザクションが十分大きな累積荷重を得たと商人がみなした後，商品を受け取る.
2. 攻撃者は二重支払いのトランザクションを発行する.
3. 攻撃者は計算能力を使って，二重支払いを承認する多数の小規模なトランザクションを発行する．しかし，このトランザクションは攻撃者が商人に送った本来のトランザクションを直接的または間接的に承認しない.
4. 攻撃者がチップを承認する必要のない多くのシビルアイデンティティを持つこと（Sybil Attack）は可能である.
5. 3 番目と違った方法としては，攻撃者がすべての計算能力を使って大きな二重支払いのトランザクションを発行することである．このトランザクションはとても

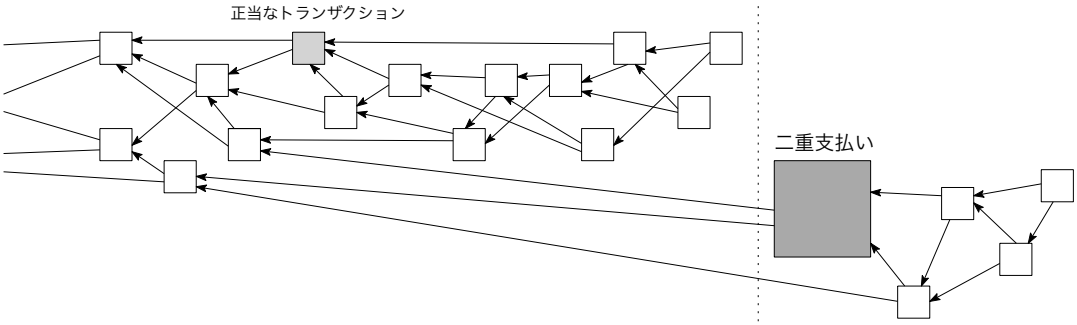

図 5: “大きい荷重”攻撃

大きな自身の荷重[19]を持ち，商人に支払うために使われた正当なトランザクションの前のトランザクションたちを承認する．

6. 攻撃者はその不正なサブ Tangle が正当なサブ Tangle を上回ることを期待する．このような状況が起こると，メイン Tangle はこの二重支払いのトランザクションから成長を続けることになり，商人への本来の支払いを有する正統な分岐は孤立する（図 5 を参照）．

実際，1 つの大きな二重支払いトランザクションの戦略によって，攻撃者が成功する可能性が高まることがわかる．この数学的モデルの“理想的な”状況では，この攻撃は**常に**成功する．

$W^{(n)}$ を二重支払いトランザクションに少なくとも 3^n の荷重を与えるナンスを得るのに必要な時間と仮定する．$W^{(n)}$ を $\mu 3^{-n}$ のパラメーター[20]を持つ指数関数的に分布する確率変数と仮定しても良い．ここで μ は攻撃者の計算能力の大きさを表す．

商人が，累積荷重が少なくとも w_0 になる時に正当なトランザクションを受け入れ，元のトランザクションから t_0 時間単位で起こると仮定する．そうすると累積荷重が線形速度 λw で成長すると期待するのは妥当である．ここで，λ は誠実なノードによってネットワーク上で発行されたトランザクションの全体的な到着率であり，w は一般的なトランザクションの荷重を意味する．この時の正統な分岐の典型的な合計荷重は $w_1 = \lambda w t_0$ である．

$\lceil x \rceil$ は x 以上の最小の整数であり，$n_0 = \left\lceil \frac{\ln w_1}{\ln 3} \right\rceil$ と定義しよう．そのため，$3^{n_0} \geq w_1$ である[21]．もし，長さ t_0 の時間間隔の間に，攻撃者が二重支払いトランザクションに少なくとも 3^{n_0} の荷重を与えるナンスを得ることが出来た場合，そのとき攻撃は成功する．

[19] ここでは，トランザクションの自身の荷重が変わる可能性があると仮定する．下の議論で自身の荷重を変えることが良いアイディアであることが明らかになる．

[20] $\mu^{-1}3^n$ の期待値を持つ．

[21] 実際に，w_1 が大きい時 $3^{n_0} \approx w_1$ である．

そのような出来事の確率は,

$$\mathbb{P}[W^{(n_0)} < t_0] = 1 - \exp(-t_0\mu 3^{-n_0}) \approx 1 - \exp(-t_0\mu w_1^{-1}) \approx \frac{t_0\mu}{w_1}.$$

この近似値は $\frac{t_0\mu}{w_1}$ が小さい場合に真であり，これは合理的な仮定である．もし，この“即座の”攻撃が成功しなかった場合，攻撃者は継続して $n > n_0$ に対して 3^n の荷重を与えるナンスを探し続け，見つかった場合は正統な分岐の合計荷重が 3^n より小さい事を期待する．この出来事が起こる確率は

$$\mathbb{P}[\lambda w W^{(n)} < 3^n] = 1 - \exp\big(-\mu 3^{-n_0} \times (3^{n_0}/\lambda w)\big) = 1 - \exp(-\mu/\lambda w) \approx \frac{\mu}{\lambda w}.$$

つまり $\frac{\mu}{\lambda w}$ は通常小さいはずであるにも関わらず，各“レベル”n において，この攻撃は一定の確率で成功する．従って，この攻撃は成功するであろう．成功するまでの典型的な時間はおおよそ $3^{\frac{\lambda w}{\mu}}$ である．この数は非常に大きいかもしれないが，“最初”[22]の攻撃が成功する可能性は無視できない．従って，対策が必要である．そのような対策の一つは，自身の荷重に上限を設けたり，一定の値に設定したりすることである．3章で言及したように，スパムからの保護が十分でないため，後者はベストな解決策ではない可能性がある．

今，最大の自身の荷重が 1 に上限設定される状況について考察し，この攻撃が成功する確率を推定しよう．

ある与えられたトランザクションが発行された瞬間から t_0 単位時間で累積重量 w_0 を獲得し，そのトランザクションの適応期間が終了したと仮定する．この状況では，トランザクションの累積荷重は速度 λ で直線的に増加する．今，攻撃者はこのトランザクションに二重支払いを仕掛けたいと考えよう．そうするためには，攻撃者は秘密裏に二重支払いトランザクションを準備し，商人への本来のトランザクションが発行された時点[23]で，二重支払いトランザクションを承認する**無意味な**トランザクションの生成を始める．商人が正当なトランザクションを受け入れることを決めた後，しばらくして攻撃者のサブ Tangle が正当なサブ Tangle を上回ると，二重支払い攻撃は成功する．もし上回ることがなければ，二重支払いトランザクションは他のトランザクションから承認されない．なぜなら，正当なトランザクションがより多くの累積荷重を獲得し，ほとんどすべての新しいチップが間接的に正当なトランザクションを承認するからである．このシナリオでは二重支払いトランザクションは孤立する．

前と同様に，μ を攻撃者の計算能力とする．また，トランザクションが即座に伝搬すると平易に仮定する．$G_1, G_2, G_3, \ldots$ はパラメータ μ[24]を持つ独立同分布（i.i.d.）な指数確率変数とし，$V_k = \mu G_k$，$k \geq 1$ とする．従って $V_1, V_2, V_3, \ldots$ はパラメータ 1 を持つ独立同分布（i.i.d.）な指数確率変数となる．

[22]時間 t_0 の間.
[23]または前にも．後でこの場合について考察する.
[24]期待値 $1/\mu$ を持つ.

t_0 の時点で商人が累積荷重 $w0$ を持つトランザクションを受け入れると仮定する．攻撃者が二重支払いに成功する確率を推定しよう．$M(\theta) = (1-\theta)^{-1}$ をパラメータ 1 を持つ指数分布の積率母関数とする（[14] の 7.7 節を参照）．これは $\alpha \in (0,1)$ において以下が成り立つことが知られている[25]．

$$\mathbb{P}\Big[\sum_{k=1}^{n} V_k \le \alpha n\Big] \approx \exp\big(-n\varphi(\alpha)\big). \tag{9}$$

ここで $\varphi(\alpha) = -\ln\alpha + \alpha - 1$ は $\ln M(\theta)$ のルジャンドル変換である．一般的な事実として，$\alpha \in (0,1)$ において $\varphi(\alpha) > 0$ が成り立つ．パラメータが 1 の指数分布に従う確率変数の期待値が 1 に等しいことも思い出していただきたい．

また $\frac{\mu t_0}{w_0} < 1$ も仮定する．でないと，攻撃者のサブ Tangle が最終的に正当な Tangle を上回る確率が 1 に近くなる．今ここで，t_0 の時点で w_0 を上回るには，攻撃者は最低でも t_0 の間に自身の最大荷重が m の w_0 を発行することが可能でなければならない．従って，式 (9) を用いて，二重支払いのトランザクションが t_0 の時点でより大きい累積荷重を持つ確率はおおよそ以下であることが得られる．

$$\begin{aligned}
\mathbb{P}\Big[\sum_{k=1}^{w_0/m} G_k < t_0\Big] &= \mathbb{P}\Big[\sum_{k=1}^{w_0} V_k < \mu t_0\Big] \\
&= \mathbb{P}\Big[\sum_{k=1}^{w_0} V_k < w_0 \times \frac{\mu t_0}{w_0}\Big] \\
&\approx \exp\big(-w_0\varphi\big(\tfrac{\mu t_0}{w_0}\big)\big).
\end{aligned} \tag{10}$$

上の確率が小さくなるためには，$\frac{w_0}{m}$ が大きくなければならず，$\varphi\big(\frac{\mu t_0}{w_0}\big)$ がそれほど小さくない必要がある．

$t \ge t_0$ の時，正当なトランザクションの累積荷重がおおよそ $w_0 + \lambda(t - t_0)$ であることに注意されたい．なぜなら，適応期間が終わると累積荷重が λ のスピードで成長すると仮定したからである．式 (10) と類似して，二重支払いのトランザクションが時間 $t \ge t_0$ の時，もっと大きい累積荷重を持つ確率についておおよそ次が得られる．

$$\exp\big(-(w_0 + \lambda(t-t_0))\varphi\big(\tfrac{\mu t}{w_0+\lambda(t-t_0)}\big)\big). \tag{11}$$

その時，適応期間中は累積荷重は λ 未満のスピードで成長するので，$\frac{\mu t_0}{w_0} \ge \frac{\mu}{\lambda}$ であることは真でなくてはならない．二重支払いが成功する確率は次とみなせる．

$$\exp\big(-w_0\varphi\big(\max(\tfrac{\mu t_0}{w_0}, \tfrac{\mu}{\lambda})\big)\big). \tag{12}$$

[25] これはいわゆる大偏差原理の結果である．上界の簡単でためになる導出には一般書 [13] に加え，[14] の 8.5 節の命題 1.9 を，下界の（それほど簡単でない）導出には [5] の 1.9 節を参照のこと．

例えば，$\mu = 2, \lambda = 3$ の時，攻撃者の力は残りのネットワークの力よりわずかに少なくなる．トランザクションは時間 12 までに累積荷重 32 を得ると仮定する．この時，$\max(\frac{\mu t_0}{w_0}, \frac{\mu}{\lambda}) = \frac{3}{4}$ であり，$\varphi\left(\frac{3}{4}\right) \approx 0.03768$ となり，そして式 (12) は約 0.29 の上界を与える．しかし，もし $\mu = 1$ でその他のすべてのパラメータはそのままと仮定すると，$\max(\frac{\mu t_0}{w_0}, \frac{\mu}{\lambda}) = \frac{3}{8}$ であり，$\varphi\left(\frac{3}{8}\right) \approx 0.3558$ となり，式 (12) は約 0.00001135 の上界を与える．これは実に激烈な変化である．

上記の議論から，システムが安全であるためには，不等式 $\lambda > \mu$ が真であるべきことを認識することは重要である．言い換えれば，“正直な”トランザクションの入力フローは攻撃者の計算能力と比べて大きくあるべきである．もしそうでなければ，式 (12) の概算は無用のものとなる．これは Tangle ベースシステムの初期段階におけて，チェックポイントなどの追加のセキュリティ対策の必要性を指摘するものである．

2 つの矛盾するトランザクションのどちらが有効かを決定する戦略を選択する時，累積荷重を決定基準として使用する場合には注意が必要である．累積荷重が 4.1 節で述べたものと同様の攻撃を受ける可能性があるという事実による．つまり攻撃者は前もって二重支払いトランザクションを十分に準備しておくことができ，その二重支払いトランザクションを参照する秘密のサブ Tangle を作成でき，商人が正当なトランザクションを受け入れた後にその秘密のサブ Tangle をばらまくことができるからである．2 つの矛盾するトランザクション間で決めるもっと良い方法は，次の章で記述されるようなものかも知れない：チップ選択アルゴリズムを実行し，2 つの矛盾するトランザクションのどちらがある選択されたチップにより間接的に承認されるかを見ることである．

4.1 パラサイトチェーン攻撃と新しいチップ選択アルゴリズム

次の攻撃を考えてみよう (図 6)：攻撃者が，メイン Tangle をたまに参照しより高いスコアを得るサブタンブルを秘密裏に構築する．信頼できるチップのスコアがおおよそメインの Tangle にある全ての自身の荷重の合計である一方，攻撃者のチップのスコアもおおよそパラサイトチェーンにある全ての自身の荷重の合計であることに注意されたい．独力でサブ Tangle を構築する攻撃者にとってネットワークの通信遅延は問題とならない[26]ので，もし攻撃者が十分に強いコンピュータを使うと，パサライト・チップにもっと多くの高さを与えることができる．さらに，攻撃者は，攻撃の瞬間に，パラサイトチェーンで以前に発行したトランザクションを承認するたくさんの新しいトランザクションをばらまくことで，人工的に攻撃者のチップ数を増やすことができる（図 6 を参照）．これは誠実なノードが，利用可能なチップ間での簡単な選択を含むいつくかの選択戦略を使う場合に，攻撃者に利点を与える．

このパラサイトチェーン攻撃に対して防御するために，メイン Tangle が攻撃者よりもより活発なハッシュパワーを持つことになっているという事実を使う．ゆえに，

[26] これは攻撃者が常に自身のトランザクションを他のネットワークの情報に頼ることなく承認できるという事実による

メイン Tangle は累積荷重においてより多くのトランザクションのために攻撃者よりも大きな増加を生み出すことができる．アイディアは参照する 2 つのチップを選ぶ時に MCMC（マルコフ連鎖モンテカルロ）アルゴリズムを用いることである．

$\mathcal{H}_x$ をあるサイトの現在の累積荷重とする．ここで全ての自身の荷重を 1 としたことを思い出していただきたい．ゆえに，チップの累積荷重は常に 1 で，他のサイトの累積荷重は少なくとも 2 である．

このアイディアはいくつかの粒子（別名ランダムウォーカーたち）を Tangle のサイト上に配置し，チップに向かってランダムに[27]歩かせるというものである．このランダムウォーカーたちによって“選ばれた”チップたちは承認の候補者となる．アルゴリズムは次のように記述される：

1. $[W, 2W]$ の間にあるすべてのサイトを考える．ここで W は妥当な大きさ[28]のものである．

2. ある間隔[29]で N 個のランダムウォーカーたちを独立的に配置する．

3. これらのランダムウォーカーたちが独立した離散時間ランダムウォークを実行するようにする．これは x から y への移行は y が x を承認した時のみ可能であることを意味する．

4. 最初にチップ集合に到達する 2 つのランダムウォーカーは承認される 2 つのチップに位置する．しかし，このルールを次のように変更することが賢明かもしれない．初めに**あまりにも速く**チップたちに到達したランダムウォーカーたちは“怠惰なチップたち”の 1 つに行き着いたかもしれないから放棄する．

5. ランダムウォーカーたちが移行する確率は次のように定義される：y が x を承認する（これを $y \rightsquigarrow x$ と表示する）場合，移行する確率 P_{xy} は $\exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_y)\big)$ に比例し，これは

$$P_{xy} = \exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_y)\big)\Big(\sum_{z:z\rightsquigarrow x} \exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_z)\big)\Big)^{-1}. \tag{13}$$

[27]ランダム性の“基準となる”ソースはない．各ノードはランダムウォーカーたちをシュミレートするために自身の（擬似）乱数生成プログラムを使用するだけである．

[28]このアイディアはランダムウォーカーたちを直ぐにチップに到着しないように Tangle の“深部に”配置することである．しかし，ランダムウォーカーたちは“すごい深部”に配置されるべきではなく，これは妥当な時間内にチップを見つける必要があるためである．また，$[W, 2W]$ は恣意的なものであり，$[W, 5W]$ ともとることができる．ランダムウォーカーたちの出発点を選択する他の方法もある．例えば，ノードは過去の t_0 から $2t_0$ までの時間単位の間に受け取られたランダムなトランザクションを単に取ることができる．ここで t_0 はある固定の時点である．

[29]この選択は主として恣意的なものである．追加のセキュリティ処置のために 2 つではなくいくつかのランダムウォーカーたちを用いる．このアイディアはランダムウォーカーたちが誤って長いと思われる攻撃者のチェーンにジャンプすると，そこで多くの時間を費やし，その間に他のチップが最初に選ばれるというものである．

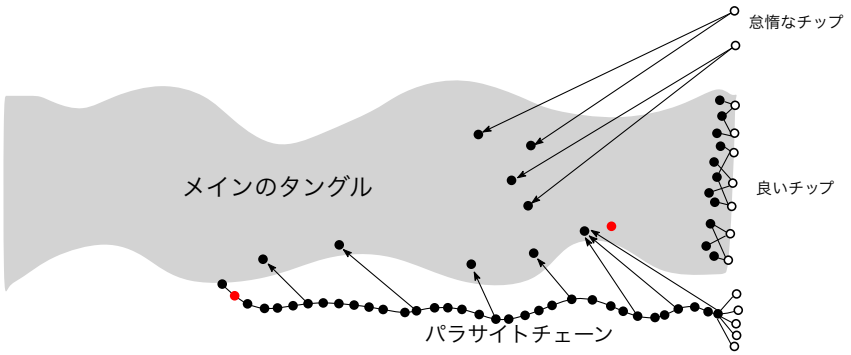

図 6: 誠実なチップやパラサイトチェーンのチップ選択アルゴリズムの視覚的描写．2 つの赤い丸は攻撃者による二重支払いの試みを示す．

ここで，$\alpha > 0$ は選択されるパラメーターである[30]．

このアルゴリズムが“ローカル”であることにご注意いただきたい．つまり，関連する計算を実行するために，ジェネシスまで Tangle を戻る必要はないということである．特に，Tangle 全体の累積荷重を計算する必要がないことに気づく．多くても，ランダムウォーカーの出発点を間接的に承認するサイトたちの累積荷重を計算する必要がある．

このアルゴリズムが意図された通りに機能するのを確認するために，まず“怠惰なチップ”を考える．怠惰なチップは確認作業を避けるためにいくつかの古いトランザクションを意図的に承認する（図 6 を参照）．ランダムウォーカーが怠惰なチップによって承認されたサイトにあったとしても，累積荷重の差が非常に大きく，P_{xy} が小さくなるため，怠惰なチップが選択される可能性は低い．

次に，別の攻撃スタイルを考える：攻撃者が秘密裏に，攻撃者のアカウントの残高が空のトランザクションを含むチェーンを攻撃者の支配下にある別のアカウント（図 6 の最も左側の赤丸で示される）に作る．そして，攻撃者はメイン Tangle で最も右側の赤丸で示されるトランザクションを発行し，商人がそのトランザクションを受け入れることを待つ．パラサイトチェーンは時折メイン Tangle を参照する．しかし，パラサイトチェーンの累積荷重はそれほどは大きくない．パラサイトチェーンは商人のトランザクションの後にはメイン Tangle を参照することができないことにご注意いただきたい．さらに，攻撃者は攻撃の瞬間に，人工的にパラサイトチェーンにあるチップの数を膨らまそうとするかもしれない（図 6 参照）．攻撃者のアイディアは新しいトランザクションたちを発行しているノードたちにパラサイトチェーンを参照させて，Tangle の誠実な分岐を孤立させることである．

MCMC 選択アルゴリズムが，高い確率で攻撃者のチップの一つを選択しない理由は理解に難くないであろう．その理由は怠惰なチップのシナリオの時と同じである．パラサイトチェーンのサイトたちはメイン Tangle 上の参照するサイトたちよりもはるか

[30] $\alpha = 1$ から始めることが出来る．

に小さい累積荷重を持つであろうからである．従って，ランダムウォーカーがパラサイトチェーンで始まらない限りは，パラサイトチェーンにジャンプする可能性は低い．さらに，メイン Tangle はパラサイトチェーンより多くのサイトたちを含んでいるので，ランダムウォーカーがパラサイトチェーンにジャンプする可能性はとても低い．

追加の保護手段として，“モデルチップ”を選択するために，はじめに大きな α（これは実際には“ほぼ決定論的”である）でランダムウォークを実行することができる．そして，実際にチップを選択するには小さな α のランダムウォークを使用するが，(間接的に）参照されるトランザクションがモデルチップと一致するかどうかを確認する．

また，チップに向かって**必ず**移動するランダムウォークでは，単純な反復を用いて，終了の確率分布を計算するのは非常に簡単で迅速であることに気づいて頂きたい．これはノードにして欲しくないことである．(ノードにして欲しいことは終了の確率分布を計算して，どのチップが“より良い”チップなのかを決めるのではなく、ランダムウォークをシミュレートしてもらいたいのである．なぜなら，終了の確率分布を用いてより良いチップを選択するとその結果を攻撃者が用いて割と簡単にアタックすることができるからである．ゆえに攻撃者が簡単に攻撃できないようにランダムウォークを用いる.）しかし，私たちのアプローチを次のように変更することは可能である：各ステップにおいて，ランダムウォークは確率（例えば）$\frac{1}{3}$ で後戻りする（すなわち，チップから 1 ステップ離れる）ことができる（そして残りの確率 $\frac{2}{3}$ を従来通りどこに向かうか分けることができる.）．とにかくランダムウォークは非常に迅速にチップに到達する（なぜならチップに向かって漂うから）が，終了手法を計算するのは本当に簡単ではない．

なぜノードたちがこのアルゴリズムに従うのかについてコメントしよう．1 章より，少なくともネットワークのノードたちの“かなりの”部分が**参照**アルゴリズムに従うと仮定することが妥当である事を思い出していただきたい．また，計算処理とネットワークの通信遅延のため，チップ選択アルゴリズムはむしろ，トランザクションが発行される瞬間まで，Tangle の過去のスナップショットを重視して機能するだろう．あとで説明する理由から，参照アルゴリズムでは，**このスナップショットをもっと過去**[31]**の時点に意図的に移すことを勧める**．攻撃者のトランザクションがすぐに承認される可能性を最大限に望む“利己的な”ノードを想像していただきたい．かなりの割合のノードたちによって採用された本章の MCMC アルゴリズムは，チップ集合における確率分布を定義する．利己的なノードの自然な最初の選択は攻撃者への配分が最大限になるチップたちを選択することであることは明らかである．しかし，たくさんの他のノードたちも利己的に振る舞い，同じ戦略を使う（これは妥当な仮定である.）と，全員が負けることになる．**多くの**新たなトランザクションがほぼ同じ時間に同じ 2 つのチップを承認するため，その後の承認のためにそれらの間であまりにも多くの競争を引き起こす．ノードたちが過去のスナップショットを使用しているので，この大量の同じ 2 つのチップを承認することよって引き起こされる累積荷重の増加をノードた

[31] 初めにランダムウォーカーはこのスナップショットを重視してかつてのチップを見つけ，それから現在の Tangle 上の“実際の”チップに向かって歩き続ける．

ちがすぐに“感じる”ことはないことも明らかである．そのため，利己的なノードであっても，参照チップ選択アルゴリズムによって生成されるデフォルトの確率分布に近い，チップ選択のための確率分布を有する，いくかのランダムなチップ承認アルゴリズム[32]を使用しなければならない．私たちはこの“塊状の”確率分布が，利己的なノードたちが存在する中で，デフォルトの確率分布と等しいとは主張しない．しかし，上記の議論はこの塊状の確率分布がデフォルトの確率分布に近いべきであることを示す．これは多くのノードたちが同じ“悪い”チップたちを検証しようとする確率を小さいままに留めることを意味する．いずれの場合も，ノードたちが利己的であろうとする大きなインセンティブはない．なぜなら，起こりうる利得が確認時間のわずかな減少にしかならないからである．これは Bitcoin のような分散型構造とは本質的に異なっている．重要な事実は，ノードたちが MCMC チップ選択アルゴリズムを放棄する理由がないことである．

式 (13) 与えられた移行確率の定義が不変であるというわけではない．指数の代わりに，$f(s) = s^{-3}$ のような急速に減少する別の関数を使うこともできる．W や N も自由に選ぶことができる．現時点では，これらのパラメータをどのように選択すべきかを正確に示す理論的な議論があるかどうかは不明である．まとめると，この章の主な貢献はチップ選択に MCMC を用いるというアイディアである．

4.2 分裂攻撃

Aviv Zohar は提案された MCMC アルゴリズムに対して以下の攻撃計画を提案した．高負荷の状態において，攻撃者は Tangle を 2 つに分裂させて，両方の間でバランスを維持しようとする．これにより 2 つの分岐は成長し続けることが可能になる．攻撃者は，誠実なノードが同時に 2 つの分岐を参照することで効果的に 2 つの分岐を結合させるのを防ぐために，少なくとも 2 つの矛盾するトランザクションを分裂の開始時に配置する必要がある．次に，攻撃者はネットワークの約半数がそれぞれの分岐に貢献することによって，攻撃者が比較的わずかな個人的な計算能力でもランダムな変動を“補正”できることを期待する．この手法が有効な場合，攻撃者は 2 つの分岐で同じ資金を使うことができる．

このような攻撃から防御するには，2 つの分岐間でバランスを維持することがとても難しい“厳しい閾値”ルールを使う必要がある．このようなルールの例は，Bitcoin ネットワークにおける最長のチェーンを選択するルールである．このコンセプトを Tangle が分裂攻撃を受けている時へと翻訳してみよう．1 つ目の分岐の合計荷重が 537，2 つ目の分岐の合計荷重が 528 と仮定する．誠実なノードが 1/2 にとても近い確率で 1 つ

[32]前に注目したように，後戻りをするために，MCMC を何度も実行する以外に，より良いチップたちを簡単に見つける（つまり“誠実な”ノードたちによって選択される可能性が高い）方法はないように思われる．しかし，MCMC を何度も実行するには時間と他のリソースが必要となる；MCMC の実行に幾らかの時間を費やすと Tangle の状態は既に変化しているので，おそらく新たに開始する必要がある．少なくとも他のノードたちのかなりの割合がデフォルトのチップ選択戦略に従うと仮定すれば，これはノードたちが MCMC チップ選択戦略を放棄して他の戦略を支持する理由がないことを説明する．

目の分岐を選んだとすると，攻撃者はおそらく 2 つの分岐間でバランスを維持することができるだろう．しかし，誠実なノードが 1/2 よりもかなり大きな確率で 1 つ目の分岐を選んだとすると，攻撃者はおそらく 2 つの分岐間でバランスを維持することができないだろう．後者の場合，2 つの分岐間でバランスを維持することができないのは，避けられないランダムな変動の後，ネットワークはすぐに 1 つの分岐を選択し，もう一方を放棄するという事実によるものである．MCMC アルゴリズムをこのように動作させるためには，非常に早く減退する関数 f を選択し，分岐が始まる前にランダムウォーカーが歩いている可能性が高くなるように，大きな深さを持つノードでランダム・ウォークを開始する必要がある．この場合，競合する分岐間の累積荷重の差が小さい場合でも，ランダム・ウォークは高い確率で“より重い”分岐を選ぶ．

ネットワーク同期化の各問題もためもあり，攻撃者の作業は非常に困難なものであることは指摘しておく価値がある：攻撃者は最近発行された多数のトランザクションを認識していないかもしれない[33]．分裂攻撃に対する防御のためのもう 1 つの効果的な方法は，十分に強力な実在物が 1 つの分岐に即時に大量のトランザクションを公開することで，パワーバランスを急速に変更させ，攻撃者がこの変更に対処するのを難しくすることである．もし，攻撃者が分裂を維持できた場合，最新のトランザクションは約 50%の確認の信頼度しか持たず（1 章参照），この分岐たちは成長しない．このシナリオでは，“誠実な”ノードたちは，分裂した分岐での矛盾するトランザクションを承認する機会を迂回して，分岐前に発生したトランザクションに対して選択的に承認を始めることを決めることができる．

チップ選択アルゴリズムの他のヴァージョンを検討することもできる．例えば，あるノードが 2 つの大きなサブ Tangle を見たとき，上記で概説した MCMC チップ選択アルゴリズムを実行する前に，自身の荷重の合計がより大きな方を選ぶ方法である．

次のアイディアも将来の実装のために検討する価値がある．式 (13) で定義された移行確率を $\mathcal{H}_x - \mathcal{H}_y$ と $\mathcal{H}_x$ の両方に依存させることで，ランダムウォーカーが Tangle の深くにいるときには，マルコフ連鎖の次のステップをほぼ決定論的にすることができ，チップに近づいたときには，よりランダムにすることができる．これは承認する 2 つのチップを選択するときに，弱い分岐に入ることを避けつつ，十分なランダム性を保証するのに役立つ．

結論：

1. 攻撃者がシステムを“凌ぐ”ことによって，二重支払いを試みたときの攻撃戦略を検討した．

2. “大きな荷重”攻撃とは，二重支払いを行うために，攻撃者は二重支払いのトランザクションにとても大きな荷重を与えようと試み，正当なサブ Tangle を上回るようにすることを意味する．この戦略は自身の荷重に制限がされていない場合

[33] “実際の”累積荷重は，攻撃者が信じる累積荷重とはかなり異なるかもしれない．

に，ネットワークの脅威になる．解決策としては，トランザクションの自身の荷重に上限を設けるか，一定値に設置することである．

3. トランザクションの最大の自身の荷重が m である状況において，攻撃者にとって最良な戦略は，二重支払いトランザクションを参照する自身の荷重が m のトランザクションたちを生成することである．“誠実な”トランザクションの入力フローが攻撃者の計算能力と比較して十分に大きい場合，式 (12) を用いて二重支払いトランザクションの累積荷重が大きくなる確率を推定することができる（式 (12) 以下の各例も参照のこと）．

4. “パラサイトチェーン”を構築する攻撃方法は，高さやスコアに基づいた承認戦略を廃止させる．なぜなら，攻撃者のサイトたちは正当な Tangle に比べて高さやスコアの値をより高くすることができるからである．一方，4.1 節で説明した MCMC チップ選択アルゴリズムはこの種の攻撃に対して保護を提供すると思われる．

5. MCMC チップ選択アルゴリズムはボーナスとして怠惰なノードたちに対する保護をも提供する．

5 量子計算への耐性

非常に大きな量子コンピュータ[34]は解を見つけるために試行錯誤に頼った問題を処理するのに非常に有効であることが知られている．Bitcoin ブロックを生成するためにナンスを見つけるプロセスは，このような問題の良い例である．今日現在では，新しいブロックを生成するための適切なハッシュを見つけるために平均で 2^{68} 回ナンスをチェックしなければならない．量子コンピュータが上記の Bitcoin パズルに類似した問題を解くために $\Theta(\sqrt{N})$ 回の演算を必要とすることが知られている（例えば [15] を参照）．この同じ問題は古典的なコンピュータでは $\Theta(N)$ 回の演算を必要とする．ゆえに，量子コンピュータは，古典的なコンピュータよりも Bitcoin ブロックチェーンマイニングにおいて約 $\sqrt{2^{68}} = 2^{34} \approx 170$ 億倍も効率的である．また，ブロックチェーンがハッシュパワーの増加に対応して難易度を増加させなかった場合，孤立したブロックが増加することも言及に値する．

同様の理由で，“大きな荷重”攻撃は，量子コンピュータによってはるかに効率的である．しかし，4 章で提案したように，荷重に上限を設けることで，量子コンピュータの攻撃をも効果的に防ぐことができる．これは明らかなことであり，理由は，iota ではトランザクションを発行するのに適したハッシュを見つけるためにチェックする必要のあるナンスの数が不当に大きくないためである．これは平均して約 3^8 である．従って，“理想的な”量子コンピュータが得られる効率の良さは，$3^4 = 81$ の位数とな

[34] 今日現在ではまだ仮説的な構造である．

り，これはすでに容認できるものである[35]．さらに重要なことには，iota の実装で使用されているアルゴリズムは，ナンスを見つける時間が，トランザクションを発行するのに必要なその他のタスクたちに必要な時間と比べてさほど大きくないようになっている．後者の部分は量子計算に対してはるかに耐性があり，ゆえに，(Bitcoin) ブロックチェーンと比較した場合，量子コンピュータを有する敵対者に対してより多くの保護を Tangle に与えるものである．

謝辞

Bartosz Kuśmierz と Cyril Grünspan と Olivia Saa と Samuel Reid と風間 徹と Rafael Kallis と Rodrigo Bueno（以上の方達は以前の WP におけるいくつかの間違いを指摘してくれた）と James Brogan（この論文をより読みやすくするために貢献してくれた）に感謝の意を表する．

参考文献

[1] Iota: a cryptocurrency for Internet-of-Things. See `http://www.iotatoken.com/`, and `https://bitcointalk.org/index.php?topic=1216479.0`

[2] bitcoinj. Working with micropayment channels. `https://bitcoinj.github.io/working-with-micropayments`

[3] PEOPLE ON NXTFORUM.ORG (2014) DAG, a generalized blockchain. `https://nxtforum.org/proof-of-stake-algorithm/dag-a-generalized-blockchain/` (registration at `nxtforum.org` required)

[4] MOSHE BABAIOFF, SHAHAR DOBZINSKI, SIGAL OREN, AVIV ZOHAR (2012) On Bitcoin and red balloons. *Proc. 13th ACM Conf. Electronic Commerce*, 56–73.

[5] RICHARD DURRETT (2004) Probability – Theory and Examples. *Duxbury advanced series.*

[6] SERGIO DEMIAN LERNER (2015) DagCoin: a cryptocurrency without blocks. `https://bitslog.wordpress.com/2015/09/11/dagcoin/`

[7] YONATAN SOMPOLINSKY, AVIV ZOHAR (2013) Accelerating Bitcoin's Transaction Processing. Fast Money Grows on Trees, Not Chains. `https://eprint.iacr.org/2013/881.pdf`

[35] $\Theta(\sqrt{N})$ は明らかに $10\sqrt{N}$ を意味することにご注意いただきたい．

[8] Yonatan Sompolinsky, Yoad Lewenberg, Aviv Zohar (2016) SPECTRE: Serialization of Proof-of-work Events: Confirming Transactions via Recursive Elections. https://eprint.iacr.org/2016/1159.pdf

[9] Yoad Lewenberg, Yonatan Sompolinsky, Aviv Zohar (2015) Inclusive Block Chain Protocols.
http://www.cs.huji.ac.il/~avivz/pubs/15/inclusive_btc.pdf

[10] Joseph Poon, Thaddeus Dryja (2016) The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments.
https://lightning.network/lightning-network-paper.pdf

[11] Sheldon M. Ross (2012) *Introduction to Probability Models.* 10th ed.

[12] David Vorick (2015) Getting rid of blocks. slides.com/davidvorick/braids

[13] Amir Dembo, Ofer Zeitouni (2010) *Large Deviations Techniques and Applications.* Springer.

[14] Sheldon M. Ross (2009) *A First Course in Probability.* 8th ed.

[15] Gilles Brassard, Peter Høyer, Alain Tapp (1998) Quantum cryptanalysis of hash and claw-free functions. *Lecture Notes in Computer Science* **1380**, 163–169.